

T7 DLL User Guide

Communication via CAN Adapter

Version 1.2

7/15/2009



US Digital • 1400 NE 136th Avenue • Vancouver, Washington • 98684 • USA •
Local: 360-260-2468 • Toll-free: 800-736-0194 • Support: 360-397-9999 • Fax: 360-260-2469
Email: info@usdigital.com • Website: www.usdigital.com

Version 1.2	1
Terms and Conditions of License for use of gratuitous software	3
1. Introduction.....	5
1.1 Purpose.....	5
1.2 Scope	5
2 Hardware Installation Instructions.....	6
3 Software Installation Instructions	6
3.1 Installing Demo Software (Windows).....	6
3.2 USB Driver Installation for CAN-USB adapter.....	6
4 Getting Started	11
4.1 Run Demo (T7 Demo).....	11
4.2 Hello-World Applications	12
5 Function Calls	16
5.1 T7_InitComm	16
5.2 T7_SetBaudRate.....	17
5.3 T7_SetMultiDropDelay.....	18
5.4 T7_CloseComm	21
5.5 T7_GetAllAngles	22
5.6 T7_GetAngle.....	23
5.7 T7_SetAngle.....	24
5.8 T7_GetAllAngleOffsets	26
5.9 T7_SetAngleOffset.....	27
5.10 T7_GetAllData	28
5.11 T7_GetAllDirections	30
5.12 T7_SetDirection.....	31
5.13 T7_GetDamping	32
5.14 T7_SetDamping.....	33
5.15 T7_GetAngleOutputRange.....	34
5.16 T7_SetAngleOutputRange.....	35
5.17 T7_GetDeviceInfo	36
5.18 T7_PingAddress	38
5.19 T7_SetAddress.....	39
6 Constants.....	42
7 Error Codes	42
8 Enumerations	43



Terms and Conditions of License for use of gratuitous software

Thank you for purchasing US Digital products.

By downloading or using US Digital software, you agree to the terms and conditions below and as further detailed on our website at <http://www.usdigital.com/company/terms-conditions.shtml>. If you do not agree with such terms and conditions, do not use the software. You may promptly return the software and other items that are part of this product in their original package with your sales receipt to your point of purchase for a full refund, or if you have downloaded this software from a US Digital web site, then you must stop using the software and destroy any copies of the software in your possession or control. These terms and conditions which accompany the original or new versions of the software and patches, point releases, maintenance releases, updates, enhancements, or upgrades thereto upon installation or download, are applicable.

Permission to use, copy, modify and distribute this software without fee is hereby granted. US Digital makes no warranty or representations about the suitability of the software for any purpose. It is provided "AS IS" without any express or implied warranty, including the implied warranties of merchantability, fitness for a particular purpose and non-infringement. US Digital shall not be liable for any direct, indirect, special or consequential damages resulting from the loss of use, data or projects, whether in an action of contract or tort, arising out of or in connection with the use or performance of this software. Your use of the software is entirely at your own risk. In connection with the software, you agree to comply with all export laws and restrictions and regulations of the Department of Commerce, the United States Department of Treasury Office of Foreign Assets Control ("OFAC"), or other United States or foreign agency or authority, and you agree not to export, or allow the export or re-export of the software in violation of any such restrictions, laws or regulations.

Downloading or using US Digital software is implicit acceptance of these terms and conditions and as further detailed at <http://www.usdigital.com/company/terms-conditions.shtml>.



US Digital • 1400 NE 136th Avenue • Vancouver, Washington • 98684 • USA •
Local: 360-260-2468 • Toll-free: 800-736-0194 • Support: 360-397-9999 • Fax: 360-260-2469
Email: info@usdigital.com • Website: www.usdigital.com

Amendments

Date	Comment(s)
07/15/2009	Added commands T7_GetMultiDropDelay and T7_SetMultiDropDelay. Version 1.2
04/08/2009	Command response now includes address, length, and command. Version 1.1
3/27/2008	CAN Adapter Host Serial Communication User Guide. Version 1.0



US Digital • 1400 NE 136th Avenue • Vancouver, Washington • 98684 • USA •
Local: 360-260-2468 • Toll-free: 800-736-0194 • Support: 360-397-9999 • Fax: 360-260-2469
Email: info@usdigital.com • Website: www.usdigital.com

1. Introduction

1.1 Purpose

This document describes how to install and use the T7 Demo software and T7 DLL on a PC running Windows. The T7 DLL provides a set of simple functions to access a network of T7 inclinometers through the CAN-232, CAN-485 or CAN-USB adapter board. The T7 DLL communicates directly with the CAN-232/CAN-485/CAN-USB adapter board using an RS232, RS485 or virtual COM port interface. The underlying serial commands used by all 3 adapter boards are identical. The CAN adapter board translates the serial port commands from the host to the USD-CAN protocol and handles all network functions to access each T7 on the network. For non-PC applications or more advanced users, the CAN adapter's serial protocol is documented in the "**CAN Adapter RS232 Communication User Guide**" document so that users can write their own routines to communicate with T7's on the network through the CAN adapter.

1.2 Scope

This document shall describe how to use each of the available interface methods provided by the T7. The following chapters are included.

- Installation Instructions
- Troubleshooting
- Getting Started
- Function Calls
- Constants
- Error Codes
- Enumerations



2 Hardware Installation Instructions

On the T7 network side, standard and terminating USD-CAN cables and T-connectors are available to wire the T7's together. There are three cabling options to choose from when it comes to host serial communication with the CAN Adapter: RS232 cable, RS485 cable, or USB. Wire the network and the CAN adapter to the host according to the information in the "**CAN-232/CAN-485/CAN-USB Datasheet**"

3 Software Installation Instructions

3.1 *Installing Demo Software (Windows)*

Download the T7 demo software from US Digital website <http://www.usdigital.com/support/software/T7-software> or insert the USD-SW CD and run the T7 Demo Installation.

The installation program will install the T7 Demo and the appropriate libraries and drivers.

3.2 *USB Driver Installation for CAN-USB adapter*

This section is applicable only if the CAN-USB adapter is being used. Connect the corresponding ends of USB cable between the CAN-USB and the host PC.

If this is the first time that a CAN-USB adapter is connected to the PC, Windows will display the **Found New Hardware Wizard** window. See Section 3.2.1 for Windows XP and Section 3.2.2 for Windows Vista.



3.2.1 Installation Instructions for Windows XP

After you have installed the T7 Software and inserted a CAN-USB device into an available USB port, Windows will display the following “Found New Hardware Wizard” dialog. Select “No, not this time” option and click “Next”.



Select “Install the software automatically (Recommended)” option and click “Next”.



Windows will automatically search for the appropriate drivers.



The USB Serial drivers included in this installation are currently being Windows Logo certified. Until certified drivers are available, please click the “Continue Anyway” button when the following Hardware Installation dialog is displayed.

Note: The Found New Hardware Wizard will appear twice as two different drivers are required to support the virtual COM port.

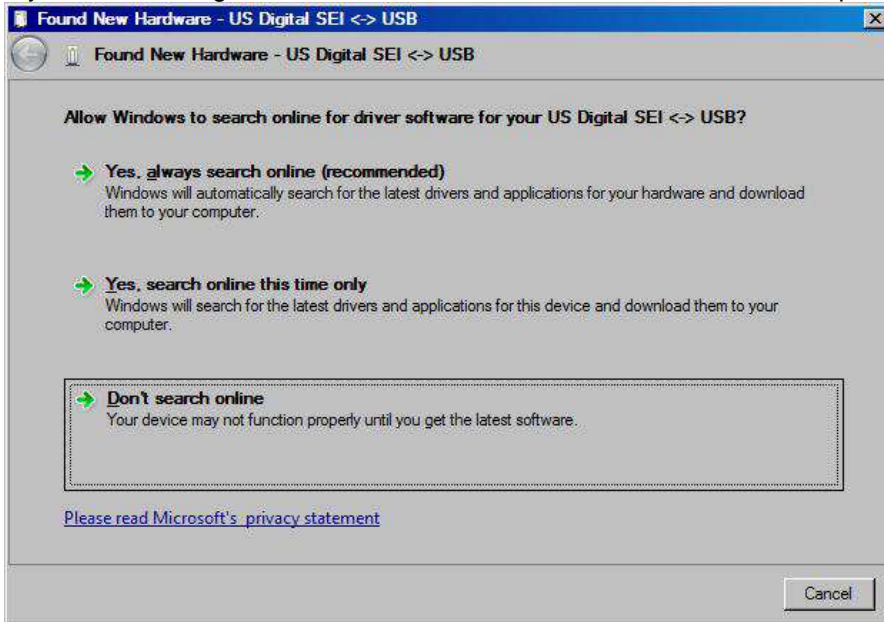


3.2.2 Installation Instructions for Windows Vista

After you have installed the T7 Software and inserted a CAN-USB device into an available USB port, Windows will display the following “Found New Hardware Wizard” dialog. Select “No, not this time” option and click “Next”.



If you are installing from a USD CD, then select “Don't search online” option.



Make sure that USD CD is inserted and click "Next". Windows will automatically search for the appropriate drivers located on a USD CD.



The USB Serial drivers included in this installation are currently being Windows Logo certified. Until certified drivers are available, please select the "Install this driver software anyway" option when the following Hardware Installation dialog is displayed.

Note: The Hardware Installation dialog will be displayed twice as two different drivers are required to support the virtual COM port.



4 Getting Started

4.1 Run Demo (T7 Demo)

After the T7 software package is installed and the T7's, CAN adapter and host PC are wired together, the T7 Demo software may be run.

From the Start | Programs | US Digital | T7 Software menu, click on the T7 Demo. The following demo application window will be displayed. Real time updates of the current angle and temperature of a selected T7 on the network are shown in the window. The T7 Demo also shows the current configuration settings of an T7 such as the damping time and offset. Changing a configuration setting will also automatically change the value in the T7's flash memory so that it will power up with the same settings.

Address	Serial #	Version	Product ID	Angle Output Range	Damping
4	28264	0.33	T7-1	-180 to 179.999	1000

Axis 0 (Pitch)	Axis 1 (Roll)	Axis 2 (Yaw)
30.735° Set 0°	0.00° Set 0°	43.368° Set 0°

Damping code set to 1000. Connected on COM1, baud = 115200 28.22°C 82.80°F

The demo provides context sensitive tool-tip help information. Simply move the mouse cursor over a user input to display tool-tip information. The source code for the T7 Demo is included in the T7 software installation.

4.2 Hello-World Applications

This example shows how easy it is to start communicating with a T7 device by writing a few lines of code.

4.2.1 VB Example

1. Run Visual Basic 6.0 and create a new Standard EXE project.
2. Include the T7UserDeclarations.bas file in your project by Clicking on Project\Add Module. Click on the tab labeled Existing and then locate and open T7UserDeclarations.bas file.
3. Double click on the form to view the Form_Load subroutine and then paste the following code within the Form_Load sub:

```
Option Explicit

Private Sub Form_Load()
    Dim lResult          As Long
    Dim bytComPort       As Byte
    Dim bytAddress       As Byte
    Dim dblAngles(2)    As Double
    Dim dblTemperature   As Double

    ' Set the COM port number your device is attached to.
    bytComPort = 1

    ' Attempt to open the COM port.
    ' Run the T7ConfigUtility to determine which COM port you're using.
    lResult = T7_InitComm(bytComPort, e115200)

    ' Check if we're able to open the COM port.
    If T7_SUCCESS = lResult Then

        ' Identify the T7's address we want to talk too.
        bytAddress = GetAddressOfFirstDeviceFound(bytComPort)

        ' Get the angle of each axis.
        lResult = T7_GetAllAngles(bytComPort, bytAddress, dblAngles(0), dblAngles(1),
        dblAngles(2), dblTemperature)
        If lResult = T7_SUCCESS Then
            MsgBox "Axis 0 = " & Format(dblAngles(0), "0.000") & Chr(176) & vbCrLf & _
                "Axis 1 = " & Format(dblAngles(1), "0.000") & Chr(176) & vbCrLf & _
                "Axis 2 = " & Format(dblAngles(2), "0.000") & Chr(176) & vbCrLf & _
                "Temperature = " & Format(dblTemperature, "0.00") & Chr(176) & "C",
            vbInformation, "VB T7 HelloWorld - Found T7 on COM" & bytComPort & " using address " &
            bytAddress
        Else
            MsgBox "Error: " & GetT7ErrorDesc(lResult), vbExclamation, _
                "Failed to communicate with T7 on COM" & bytComPort & " using address " &
            bytAddress
        End If
    Else
        MsgBox "Error: " & GetT7ErrorDesc(lResult), vbExclamation, _
            "Failed to communicate with T7 on COM" & bytComPort
    End If

    T7_CloseComm bytComPort
    Unload Me
End Sub
```



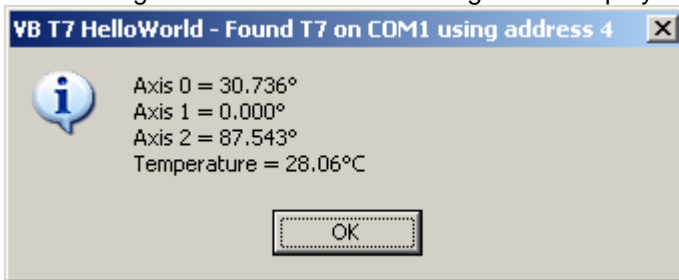
```

Private Function GetAddressOfFirstDeviceFound(bytComPort As Byte) As Byte
    Dim lResult                As Long
    Dim bytCurrentAddress(0 To 63) As Byte
    Dim bytDeviceType(0 To 63)  As Byte
    Dim ulSerial(0 To 63)       As Long
    Dim bytSize                 As Byte
    Dim i                       As Integer
    bytSize = 10

    lResult = T7_PingAddress(bytComPort, T7_BROADCAST_ADDRESS, bytCurrentAddress(0),
    bytDeviceType(0), ulSerial(0), bytSize)
    If lResult = T7_SUCCESS Then
        If bytSize > 0 Then
            GetAddressOfFirstDeviceFound = bytCurrentAddress(0)
        End If
    End If
End Function

```

- To run the application, press the F5 function key.
A message box similar to the following will be displayed.



4.2.2 C Example

1. Run Microsoft Visual C++ and create a new Win32 Console Application by clicking on File New menu items.
2. Select the Projects Tab.
3. Click Win32 Console Application and enter T7HelloWorld for the Project name. Click OK.
4. Click on the option that says, "A Hello, World! Application." and then click the Finish button and the OK button on the New Project Information dialog that pops up.
5. Copy the T7User.h and T7User.lib files to the new project directory.
6. Click on Project Settings menu items. Click on the Link tab. Add T7User.lib to the end of Object/library modules field and then click OK.
7. Click on C/C++ tab. Select Precompiled Headers from the Category drop-down menu and then select Not using precompiled headers option.
8. Click on the FileView tab within the Workspace frame. Expand the file folders and then double-click on T7HelloWorld.cpp node to open the file. Add the #include "..\Common\T7User.h" statement just above the main function.
9. Replace the code within the main function with a copy of the following code:

```
// C Hello World.cpp : Defines the entry point for the console application.
//

#include <conio.h>
#include "stdio.h"
#include "windows.h"
#include "..\Common\T7User.h"

int main()
{
    long lResult = 0;
    unsigned char ucCOMPort;
    unsigned char ucAddress = 0;
    double dblAngles[3] = {0,0,0};
    double dblTemperature = 0;

    printf("T7 HelloWorld\n");

    // Set the COM port number your device is attached to.
    ucCOMPort = 1;

    // Set the address of the T7 we want to talk to.
    ucAddress = 1;

    // Attempt to open the COM port.
    // Run the VB_T7_Demo to determine which COM port you're using.
    lResult = T7_InitComm(ucCOMPort, e115200);

    // Check if we're able to open the COM port.
    if(lResult == T7_SUCCESS)
    {
        // Get the angle of each axis.
        lResult = T7_GetAllAngles(ucCOMPort, ucAddress, &dblAngles[0], &dblAngles[1],
&dblAngles[2], &dblTemperature);
        printf("Axis 0 = %.3f\nAxis 1 = %.3f\nAxis 2 = %.3f\n", dblAngles[0], dblAngles[1],
dblAngles[2]);
    }

    if(lResult)
    {
        printf("Error: result = %d, Failed to communicate with T7 on COM%d\n", lResult,
ucCOMPort);
    }
}
```



```
    }  
    else  
    {  
        T7_CloseComm(ucCOMPort);  
    }  
  
    return 0;
```

10. } Compile the code by pressing F7 function key and run the compiled code from a DOS window so that the output may be viewed.



US Digital • 1400 NE 136th Avenue • Vancouver, Washington • 98684 • USA •
Local: 360-260-2468 • Toll-free: 800-736-0194 • Support: 360-397-9999 • Fax: 360-260-2469
Email: info@usdigital.com • Website: www.usdigital.com

5 Function Calls

For functions that need a device address, the valid range is typically 1-127. The default address for an T7 device is 127. All T7's listen to address 126.

5.1 T7_InitComm

Description:

The T7_InitComm function opens a specified COMM serial port for communication at a specified baud rate with the CAN adapter. Currently, the rate must be set to 115200 bits/sec.

Passed Parameters:

ucComPort

Specifies the COM port number to be opened. Valid range is from 1 to 255. COM port 255 is unique in that it may be used by T7_CloseComm to close all opened COM ports.

eBaudRateCode

Specifies the baud rate code

```
enum eBaudRateCode
{
    e115200 = 0, // 115200 bits/sec
    e57600 = 1,
    e38400 = 2,
    e19200 = 3,
    e9600 = 4    // 9600 bits/sec
};
```

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall T7_InitComm(unsigned char ucComPort, enum eBaudRateCode eCode);
```

Example C Usage:

```
int iResult = 0;
unsigned char ucComPort = 1;
eBaudRateCode eCode = e115200;
iResult = T7_InitComm(ucComPort, eCode);
if (iResult != T7_SUCCESS) {
    // Handle error...
}
```

VB Declaration:

```
Public Declare Function T7_InitComm Lib "T7User.dll" (ByVal bytComPort As Byte, ByVal eCode As eBaudRateCode) As Long
```



Example VB Usage:

```
Dim bytComPort As Byte
Dim IResult As Long
Dim eCode As eBaudRateCode
bytComPort = 1 ' Use COM1...
eCode = e115200
IResult = T7_InitComm(bytComPort, eCode)
If IResult <> T7_SUCCESS Then
    ' Handle error...
End If
```

5.2 T7_SetBaudRate

Description:

The T7_SetBaudRate function sets the CAN adapter to communicate at a different baud rate. Once the T7 has accepted the new baud rate, the opened COM is then updated to the new baud rate. The default baud rate is set to 115200 bits/sec.

Passed Parameters:

ucComPort

Specifies the COM port number to be accessed. Valid range is from 1 to 255.

ucAddress

Specifies the devices address. Set address = 0 for the CAN adapter.

eCode

Specifies the baud rate code.

enum eBaudRateCode

```
{
    e115200 = 0, // 115200 bits/sec
    e57600 = 1,
    e38400 = 2,
    e19200 = 3,
    e9600 = 4 // 9600 bits/sec
};
```

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall T7_SetBaudRate(unsigned char ucComPort, unsigned char ucAddress, enum eBaudRateCode eCode);
```



Example C Usage:

```
int iResult = 0;
unsigned char ucComPort = 1;
unsigned char ucAddress = 0;
eBaudRateCode eCode = e115200;
iResult = T7_SetBaudRate(ucComPort, ucAddress, eCode);
if (iResult != 0) {
    // Handle error...
}
```

VB Declaration:

```
Public Declare Function T7_SetBaudRate Lib "T7User.dll" (ByVal bytComPort As Byte, ByVal bytAddress
As Byte, ByVal eCode As eBaudRateCode) As Long
```

Example VB Usage:

```
Dim bytComPort As Byte
Dim bytAddress As Byte
Dim IResult As Long
Dim eCode As eBaudRateCode
eCode = e115200
bytComPort = 1 ' Access COM1...
bytAddress = 0
IResult = T7_SetBaudRate(bytComPort, bytAddress, eCode)
If IResult <> 0 Then
    ' Handle error...
End If
```

5.3 T7_GetMultiDropDelay

Description:

The T7_GetMultiDropDelay function is used to get the multi-drop delay.

A delay of 0 mean multi-drop mode is disabled. When multi-drop is disabled the transmitter will always be enabled. (This is normal RS-232 behavior)

A valid delay is in the range of 1 – 50 msec. This enables multi-drop and all transmitters will go tri-state.

Passed Parameters:

ucComPort

Specifies the COM port number to be accessed. Valid range is from 1 to 255.

ucAddress

Specifies the devices address. Set address = 126 to broadcast to all attached devices.

ucDelayMilliseconds

Specifies the delay to be multiplied by the device address in milliseconds that a device delays in responding to a command.

Returns:



US Digital • 1400 NE 136th Avenue • Vancouver, Washington • 98684 • USA •
Local: 360-260-2468 • Toll-free: 800-736-0194 • Support: 360-397-9999 • Fax: 360-260-2469
Email: info@usdigital.com • Website: www.usdigital.com

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall T7_GetMultiDropDelay (unsigned char ucComPort, unsigned char ucAddress,  
unsigned char ucDelayMilliseconds);
```

Example C Usage:

```
int iResult = 0;  
unsigned char ucComPort = 1;  
unsigned char ucAddress = 0;  
unsigned char ucDelayMilliseconds = 0;  
iResult = T7_GetMultiDropDelay(ucComPort, ucAddress, &ucDelayMilliseconds);  
if (iResult != 0) {  
    // Handle error...  
}
```

VB Declaration:

```
Public Declare Function T7_GetMultiDropDelay Lib "T7User.dll" (ByVal bytComPort As Byte, ByVal  
bytAddress As Byte, ByVal bytDelayMilliseconds As Byte) As Long
```

Example VB Usage:

```
Dim bytComPort As Byte  
Dim bytAddress As Byte  
Dim IResult As Long  
Dim bytDelayMilliseconds As Byte  
bytDelayMilliseconds = 0  
bytComPort = 1 ' Access COM1...  
bytAddress = 0  
IResult = T7_GetMultiDropDelay(bytComPort, bytAddress, bytDelayMilliseconds)  
If IResult <> 0 Then  
    ' Handle error...  
End If
```

5.4 T7_SetMultiDropDelay

Description:

The T7_SetMultiDropDelay function is used to enable or disable multi-drop support. This command should be sent to all devices so that they are all configured the same.

A delay of 0 disables multi-drop. When multi-drop is disabled the transmitter will always be enabled. (This is normal RS-232 behavior)

A valid delay is in the range of 1 – 50 msec. This enables multi-drop and all transmitters will go tri-state.

Passed Parameters:

ucComPort

Specifies the COM port number to be accessed. Valid range is from 1 to 255.

ucAddress

Specifies the devices address. Set address = 126 to broadcast to all attached devices.



ucDelayMilliseconds

Specifies the delay to be multiplied by the device address in milliseconds that a device delays in responding to a command.

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall T7_SetMultiDropDelay(unsigned char ucComPort, unsigned char ucAddress,  
unsigned char ucDelayMilliseconds);
```



US Digital • 1400 NE 136th Avenue • Vancouver, Washington • 98684 • USA •
Local: 360-260-2468 • Toll-free: 800-736-0194 • Support: 360-397-9999 • Fax: 360-260-2469
Email: info@usdigital.com • Website: www.usdigital.com

Example C Usage:

```
int iResult = 0;
unsigned char ucComPort = 1;
unsigned char ucAddress = 0;
unsigned char ucDelayMilliseconds = 1; ' Nominal value = 115200 / baudrate
iResult = T7_SetMultiDropDelay(ucComPort, ucAddress, ucDelayMilliseconds);
if (iResult != 0) {
    // Handle error...
}
```

VB Declaration:

```
Public Declare Function T7_SetMultiDropDelay Lib "T7User.dll" (ByVal bytComPort As Byte, ByVal bytAddress As Byte, ByVal bytDelayMilliseconds As Byte) As Long
```

Example VB Usage:

```
Dim bytComPort As Byte
Dim bytAddress As Byte
Dim IResult As Long
Dim bytDelayMilliseconds As Byte
bytDelayMilliseconds = 1
bytComPort = 1 ' Access COM1...
bytAddress = 0
IResult = T7_SetBaudRate(bytComPort, bytAddress, bytDelayMilliseconds)
If IResult <> 0 Then
    ' Handle error...
End If
```

5.5 T7_CloseComm

Description:

The T7_CloseComm function closes a specified open COM port.

Passed Parameters:

ucComPort

Specifies the COM port number to be closed. Valid range is from 1 to 255. A value of 255 will cause all opened COM ports to close.

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall T7_CloseComm(unsigned char ucComPort);
```



Example C Usage:

```
int iResult = 0;
unsigned char ucComPort = 1;
iResult = T7_CloseComm(ucComPort);
if (iResult != 0) {
    // Handle error...
}
```

VB Declaration:

```
Public Declare Function T7_CloseComm Lib "T7User.dll" (ByVal bytComPort As Byte) As Long
```

Example VB Usage:

```
Dim bytComPort As Byte
Dim IResult As Long
bytComPort = 1 ' Close COM1...
IResult = T7_CloseComm(bytComPort)
If IResult <> T7_SUCCESS Then
    ' Handle error...
End If
```

5.6 T7_GetAllAngles

Description:

The T7_GetAllAngles function retrieves the angle (in degrees) for each axis. See T7_GetAngleOutputRange to determine if the range of values returned are bidirectional (-180.000 to 179.999) or unidirectional (0.000 to 359.999).

Passed Parameters:

ucComPort

Specifies the COM port number to be accessed. Valid range is from 1 to 255.

ucAddress

Specifies the devices address. Valid range is from 1 to 127.

pdblAngle0

Pointer to a double value which will receive angle for axis 0.

pdblAngle1

Pointer to a double value which will receive angle for axis 1.

pdblAngle2

Pointer to a double value which will receive angle for axis 2.

pdblTemperature

Pointer to a double value which will receive the devices temperature in Celsius (-40.00 to 150.00).

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall T7_GetAllAngles(unsigned char ucComPort, unsigned char ucAddress, double *  
pdblAngle0, double * pdblAngle1, double * pdblAngle2, double * pdblTemperautre);
```



Example C Usage:

```
int iResult = 0;
unsigned char ucComPort = 1;
unsigned char ucAddress = 1;
double dblAngles[3] = {0,0,0};
double dblTemperature = 0;
iResult = T7_GetAllAngles(ucComPort, ucAddress, &dblAngles[0], &dblAngles[1], &dblAngles[2],
&dblTemperature);
if (iResult != T7_SUCCESS) {
    // Handle error...
}
```

VB Declaration:

```
Public Declare Function T7_GetAllAngles Lib "T7User.dll" (ByVal bytComPort As Byte, ByVal bytAddress
As Byte, ByRef dblAngle0 As Double, ByRef dblAngle1 As Double, ByRef dblAngle2 As Double, ByRef
dblTemperature As Double) As Long
```

Example VB Usage:

```
Dim bytComPort As Byte
Dim bytAddress As Byte
Dim IResult As Long
Dim dblAngle0 As Double
Dim dblAngle1 As Double
Dim dblAngle2 As Double
Dim dblTemp As Double
```

```
bytComPort = 1 ' Access COM1...
bytAddress = 1
IResult = T7_GetAllAngles(bytComPort, bytAddress, dblAngle0, dblAngle1, dblAngle2, dblTemp)
If IResult <> T7_SUCCESS Then
    ' Handle error...
End If
```

5.7 T7_GetAngle

Description:

The T7_GetAngle function gets one angle (in degrees) for a specified axis.
See T7_GetAngleOutputRange to determine if the range of values returned are bidirectional (-180.000 to 179.999) or unidirectional (0.000 to 359.999).

Passed Parameters:

ucComPort

Specifies the COM port number to be accessed. Valid range is from 1 to 255.

ucAddress

Specifies the devices address. Valid range is from 1 to 127.

eAxis

Specifies the axis of whose angle should be returned. Valid range is from 0 to 2.

pdblAngle

Pointer to a double value which will receive the angle for the specified axis.



Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall T7_GetAngle(unsigned char ucComPort, unsigned char ucAddress, enum eAxis axis, double * pdblAngle);
```

Example C Usage:

```
int iResult = 0;
unsigned char ucComPort = 1;
unsigned char ucAddress = 1;
double dblAngle = 0;
iResult = T7_GetAngle(ucComPort, ucAddress, eZAxis, &dblAngle);
if (iResult != T7_SUCCESS) {
    // Handle error...
}
```

VB Declaration:

```
Public Declare Function T7_GetAngle Lib "T7User.dll" (ByVal bytComPort As Byte, ByVal bytAddress As Byte, ByVal axis As eAxis, ByRef dblAngle As Double) As Long
```

Example VB Usage:

```
Dim bytComPort As Byte
Dim bytAddress As Byte
Dim IResult As Long
Dim dblAngle As Double
bytComPort = 1
bytAddress = 1
dblAngle = 0
IResult = T7_GetAngle(bytComPort, bytAddress, eZAxis, dblAngle)
If IResult <> T7_SUCCESS Then
    ' Handle error...
End If
```

5.8 T7_SetAngle

Description:

The T7_SetAngle function calculates and sets the internally stored angle offset value so that the currently reported angle equals the angle parameter value specified. The angle offset value is added to the angle reported by the sensor.

Passed Parameters:

ucComPort

Specifies the COM port number to be accessed. Valid range is from 1 to 255.

ucAddress

Specifies the devices address. Valid range is from 1 to 127.

eAxis



Specifies the axis of whose angle should be returned. Valid range is from 0 to 2.

pdblAngle

Specifies the angle that should be reported for the specified axis at the current orientation. Valid range is from -180.00 to 179.999 or 0 to 359.999 degrees depending on the selected angle output range.

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall T7_SetAngle(unsigned char ucComPort, unsigned char ucAddress, enum eAxis axis, double dblAngle);
```

Example C Usage:

```
int iResult = 0;
unsigned char ucComPort = 1;
unsigned char ucAddress = 1;
double dblAngle = 45.5;
iResult = T7_SetAngle(ucComPort, ucAddress, eZAxis, dblAngle);
if (iResult != T7_SUCCESS) {
    // Handle error...
}
```

VB Declaration:

```
Public Declare Function T7_SetAngle Lib "T7User.dll" (ByVal bytComPort As Byte, ByVal bytAddress As Byte, ByVal axis As eAxis, ByVal dblAngle As Double) As Long
```

Example VB Usage:

```
Dim bytComPort As Byte
Dim bytAddress As Byte
Dim IResult As Long
Dim dblAngle As Double
bytComPort = 1
bytAddress = 1
dblAngle = 45.5
IResult = T7_SetAngle(bytComPort, bytAddress, eZAxis, dblAngle)
If IResult <> T7_SUCCESS Then
    ' Handle error...
End If
```



5.9 T7_GetAllAngleOffsets

Description:

The T7_GetAngleOffset function retrieves the angle offsets (in degrees) for all axes.

Passed Parameters:

ucComPort

Specifies the COM port number to be accessed. Valid range is from 1 to 255.

ucAddress

Specifies the devices address. Valid range is from 1 to 127.

pdblAngle0

Pointer to a double value which will receive angle offset for axis 0.

pdblAngle1

Pointer to a double value which will receive angle offset for axis 1.

pdblAngle2

Pointer to a double value which will receive angle offset for axis 2.

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall T7_GetAngleOffset(unsigned char ucComPort, unsigned char ucAddress, double * pdblOffset0, double * pdblOffset1, double * pdblOffset2);
```

Example C Usage:

```
int iResult = 0;
unsigned char ucComPort = 1;
unsigned char ucAddress = 1;
double dblOffsets[3] = {0,0,0};
iResult = T7_GetAllAngleOffsets(ucComPort, ucAddress, &dblOffsets[0], &dblOffsets[1], &dblOffsets[2]);
if (iResult != T7_SUCCESS) { // Handle error...}
```

VB Declaration:

```
Public Declare Function T7_GetAllAngleOffsets Lib "T7User.dll" (ByVal bytComPort As Byte, ByVal bytAddress As Byte, ByVal dblOffset0 As Double, ByVal dblOffset1 As Double, ByVal dblOffset2 As Double) As Long
```

Example VB Usage:

```
Dim bytComPort As Byte
Dim bytAddress As Byte
Dim IResult As Long
Dim dblOffsets(0 To 2) As Double
bytComPort = 1 ' Access COM1...
bytAddress = 1
IResult = T7_GetAllAngleOffsets(bytComPort, bytAddress, dblOffsets(0), dblOffsets(1), dblOffsets(2))
If IResult <> T7_SUCCESS Then
    ' Handle error...
```



End If

5.10 T7_SetAngleOffset

Description:

The T7_SetAngleOffset function sets the angle offset (in degrees) for a specified axis.

Passed Parameters:

ucComPort

Specifies the COM port number to be accessed. Valid range is from 1 to 255.

ucAddress

Specifies the devices address. Valid range is from 1 to 127.

eAxis

Specifies the axis of whose angle should be set. Valid range is from 0 to 2.

dblOffset

Specifies the new angle offset.

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall T7_SetAngleOffset(unsigned char ucComPort, unsigned char ucAddress, enum eAxis  
axis, double dblOffset);
```

Example C Usage:

```
int iResult = 0;  
unsigned char ucComPort = 1;  
unsigned char ucAddress = 1;  
double dblOffset = 90.0;  
iResult = T7_SetAngleOffset(ucComPort, ucAddress, eZAxis, dblOffset);  
if (iResult != T7_SUCCESS) {  
    // Handle error...  
}
```

VB Declaration:

```
Public Declare Function T7_SetAngleOffset Lib "T7User.dll" (ByVal bytComPort As Byte, ByVal  
bytAddress As Byte, ByVal axis As eAxis, ByVal dblOffset As Double) As Long
```

Example VB Usage:

```
Dim bytComPort As Byte  
Dim bytAddress As Byte  
Dim IResult As Long  
Dim dblOffset As Double  
bytComPort = 1  
bytAddress = 1  
dblOffset = 90.0  
IResult = T7_SetAngleOffset(bytComPort, bytAddress, eZAxis, dblOffset)  
If IResult <> T7_SUCCESS Then  
    ' Handle error...
```



End If

5.11 T7_GetAllData

Description:

The T7_GetAllData function retrieves the angle (in degrees) for each axis, device temperature, damped acceleration outputs for each axis, and the device's serial number.

See T7_GetAngleOutputRange to determine if the range of values returned are (-180.000 to 179.999) or (0 to 359.999).

Passed Parameters:

ucComPort

Specifies the COM port number to be accessed. Valid range is from 1 to 255.

ucAddress

Specifies the devices address. Valid range is from 1 to 127.

pdblAngle0

Pointer to a double value which will receive angle for axis 0.

pdblAngle1

Pointer to a double value which will receive angle for axis 1.

pdblAngle2

Pointer to a double value which will receive angle for axis 2.

pdblTemperature

Pointer to a double value which will receive the devices temperature in Celsius (-40.00 to 150.00).

pdblAccel0

Pointer to a double value which will receive acceleration for axis 0.

pdblAccel1

Pointer to a double value which will receive acceleration for axis 1.

pdblAccel2

Pointer to a double value which will receive acceleration for axis 2.

plSerialNo

Pointer to an unsigned long value which will receive the device's serial number.

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall T7_GetAllData(unsigned char ucComPort, unsigned char ucAddress, double *  
pdblAngle0, double * pdblAngle1, double * pdblAngle2, double * pdblTemperature, double * pdblAccel0,  
double * pdblAccel1, double * pdblAccel2, unsigned long * plSerialNo);
```



Example C Usage:

```
int iResult = 0;
unsigned char ucComPort = 1;
unsigned char ucAddress = 1;
double dblAngle[3] = {0,0,0};
double dblTemperature = 0.0;
double dblAccel[3] = {0,0,0};
unsigned long ulSerialNo = 0;
```

```
iResult = T7_ReadAllData(ucComPort, ucAddress, &dblAngle[0], &dblAngle[1], &dblAngle[2],
&dblTemperature, &dblAccel[0], &dblAccel[1], &dblAccel[2], &ulSerialNo);
if (iResult != T7_SUCCESS) { // Handle error...}
```

VB Declaration:

```
Public Declare Function T7_GetAllData Lib "T7User.dll" (ByVal bytComPort As Byte, ByVal bytAddress
As Byte, ByRef dblAngle0 As Double, ByRef dblAngle1 As Double, ByRef dblAngle2 As Double, ByRef
dblTemperature As Double, ByRef dblAccel0 As Double, ByRef dblAccel1 As Double, ByRef dblAccel2
As Double, ByRef lSerialNo As Long) As Long
```

Example VB Usage:

```
Dim bytComPort As Byte
Dim bytAddress As Byte
Dim lResult As Long
Dim dblAngles(0 To 2) As Double
Dim dblTemperature As Double
Dim dblAccels(0 To 2) As Double
Dim lSerialNo As Long
bytComPort = 1 ' Access COM1...
bytAddress = 1
```

```
lResult = T7_GetAllData(bytComPort, bytAddress, dblAngles(0), dblAngles(1), dblAngles(2),
dblTemperature, dblAccels(0), dblAccels(1), dblAccels(2), lSerialNo)
If lResult <> T7_SUCCESS Then
' Handle error...
End If
```



5.12 T7_GetAllDirections

Description:

The T7_GetAllDirections function retrieves the angle increase direction of each axis (normal or reversed).

Passed Parameters:

ucComPort

Specifies the COM port number to be accessed. Valid range is from 1 to 255.

ucAddress

Specifies the devices address. Valid range is from 1 to 127.

puDirections0

Pointer to axis 0 angle increase direction byte. 0 = normal, 1 = reversed.

puDirections1

Pointer to axis 1 angle increase direction byte. 0 = normal, 1 = reversed.

puDirections2

Pointer to axis 2 angle increase direction byte. 0 = normal, 1 = reversed.

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall T7_GetAllDirections(unsigned char ucComPort, unsigned char ucAddress, unsigned char * puDirection0, unsigned char * puDirection1, unsigned char * pcDirection2);
```

Example C Usage:

```
int iResult = 0;
unsigned char ucComPort = 1;
unsigned char ucAddress = 1;
unsigned char cDirections[3] = {0,0,0};
iResult = T7_GetAllDirections(ucComPort, ucAddress, &ucDirections[0], &ucDirections[1], &ucDirections[2]);
if (iResult != T7_SUCCESS) {
    // Handle error...
}
```

VB Declaration:

```
Public Declare Function T7_GetAllDirections Lib "T7User.dll" (ByVal bytComPort As Byte, ByVal bytAddress As Byte, ByRef bytDirection0 As Byte, ByRef bytDirection1 As Byte, ByRef bytDirection2 As Byte) As Long
```



Example VB Usage:

```
Dim bytComPort As Byte
Dim bytAddress As Byte
Dim IResult As Long
Dim bytDirecitons(0 To 2) As Byte
bytComPort = 1 ' Access COM1...
bytAddress = 1
IResult = T7_GetAllDirections(bytComPort, bytAddress, bytDirecitons(0), bytDirecitons(1),
bytDirecitons(2))
If IResult <> T7_SUCCESS Then
    ' Handle error...
End If
```

5.13 T7_SetDirection

Description:

The T7_SetDirection function sets the angle increase direction for a specified axis (normal or reversed).

Passed Parameters:

ucComPort

Specifies the COM port number to be accessed. Valid range is from 1 to 255.

ucAddress

Specifies the devices address. Valid range is from 1 to 127.

eAxis

Specifies the axis whose angle increase direction is set. Valid range is from 0 to 2.

```
enum eAxis {
    eXAxis = 0,
    ePitch = 0,
    eYAxis = 1,
    eRoll = 1,
    eZAxis = 2,
    eYaw = 2
```

```
};
```

ucDirection

The angle increase direction byte. 0 = normal, 1 = reversed.

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall T7_SetDirection(unsigned char ucComPort, unsigned char ucAddress, enum eAxis
axis, unsigned char ucDirection);
```



Example C Usage:

```
int iResult = 0;
unsigned char ucComPort = 1;
unsigned char ucAddress = 1;
unsigned char ucDirection = 1;
iResult = T7_SetDirection(ucComPort, ucAddress, eZAxis, ucDirection);
if (iResult != 0) {
    // Handle error...
}
```

VB Declaration:

```
Public Declare Function T7_SetDirection Lib "T7User.dll" (ByVal bytComPort As Byte, ByVal bytAddress
As Byte, ByVal axis As eAxis, ByVal bytDirection As Byte) As Long
```

Example VB Usage:

```
Dim bytComPort As Byte
Dim bytAddress As Byte
Dim IResult As Long
Dim bytDirection As Byte
bytComPort = 1
bytAddress = 1
bytDirection = 1
IResult = T7_SetDirection (bytComPort, bytAddress, eZAxis, bytDirection)
If IResult <> 0 Then
    ' Handle error...
End If
```

5.14 T7_GetDamping

Description:

Requests the damping time in milliseconds.

Electronic damping is achieved by averaging multiple angle readings together to reduce noise. As the damping time is increased, the angle readings are smoother and lower in noise, but the response time is slower. The number of samples averaged per reported position can be calculated by multiplying the damping time in seconds by 640.

Example: When the damping is set to 125 milliseconds, each reported position will be the average of the previous 80 samples; 250 milliseconds will be the average of the previous 160 samples. To most closely match the damping of the T7 to US Digital's optical encoder inclinometers such as the T5, T6 and A2T, specify 125 milliseconds for standard damping, and 250 milliseconds for double damping.

Passed Parameters:

ucComPort

Specifies the COM port number to be accessed. Valid range is from 1 to 255.

ucAddress

Specifies the devices address. Valid range is from 1 to 127.



puiTime

Pointer to integer that will hold the damping time in milliseconds. Valid range is from 2 to 5,000 with values 0 and 1 being reserved.

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall T7_GetDamping(unsigned char ucComPort, unsigned char ucAddress, unsigned short * puiTime);
```

Example C Usage:

```
int iResult = 0;
unsigned char ucComPort = 1;
unsigned char ucAddress = 1;
unsigned short uiTime = 0;
iResult = T7_GetDamping(ucComPort, ucAddress, &uiTime);
if (iResult != 0) { // Handle error... }
```

VB Declaration:

```
Public Declare Function T7_GetDamping Lib "T7User.dll" (ByVal bytComPort As Byte, ByVal bytAddress As Byte, ByRef iTime As Integer) As Long
```

Example VB Usage:

```
Dim bytComPort As Byte
Dim bytAddress As Byte
Dim IResult As Long
Dim iTime As Integer
bytAddress = 1
bytComPort = 1 ' Access COM1...
IResult = T7_GetDamping(bytComPort, bytAddress, iTime)
If IResult <> 0 Then
    ' Handle error...
End If
```

5.15 T7_SetDamping

Description:

Set the damping time in milliseconds.
Refer to T7_GetDamping for an expanded description.

Passed Parameters:

ucComPort

Specifies the COM port number to be accessed. Valid range is from 1 to 255.

ucAddress

Specifies the devices address. Valid range is from 1 to 127.



uiTime

The new damping time value. Valid range is from 2 (2 milliseconds) to 5,000 (5 seconds) with values 0 and 1 being reserved.

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall T7_SetDamping(unsigned char ucComPort, unsigned char ucAddress, unsigned short uiTime);
```

Example C Usage:

```
int iResult = 0;
unsigned char ucComPort = 1;
unsigned char ucAddress = 1;
unsigned short uiTime = 500;
iResult = T7_SetDamping(ucComPort, ucAddress, uiTime);
if (iResult != 0) {
    // Handle error...
}
```

VB Declaration:

Public Declare Function T7_SetDamping Lib "T7User.dll" (ByVal *bytComPort* As Byte, ByVal *bytAddress* As Byte, ByVal *iTime* As Integer) As Long

Example VB Usage:

```
Dim bytComPort As Byte
Dim bytAddress As Byte
Dim IResult As Long
Dim iTime As Integer
bytComPort = 1 ' Access COM1...
bytAddress = 1
iTime = 500
IResult = T7_SetDamping(bytComPort, bytAddress, iTime)
If IResult <> 0 Then
    ' Handle error...
End If
```

5.16 T7_GetAngleOutputRange

Description:

The T7_GetAngleOutputRange function retrieves a single byte that indicates the angle output range for all three axes.

Passed Parameters:

ucComPort



Specifies the COM port number to be accessed. Valid range is from 1 to 255.
ucAddress

Specifies the devices address. Valid range is from 1 to 127.
pucAngleOutputRange

Pointer to the angle output range byte.
0 = bidirectional (-180.000° to 179.999°)
1 = unidirectional (0.000° to 359.999°)

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall T7_GetAngleOutputRange(unsigned char ucComPort, unsigned char ucAddress, unsigned char * pucAngleOutputRange);
```

Example C Usage:

```
int iResult = 0;
unsigned char ucComPort = 1;
unsigned char ucAddress = 1;
unsigned char ucAngleOutputRange = 0;
iResult = T7_GetAngleOutputRange(ucComPort, ucAddress, &ucAngleOutputRange);
if (iResult != 0) {
    // Handle error...
}
```

VB Declaration:

```
Public Declare Function T7_GetAngleOutputRange Lib "T7User.dll" (ByVal bytComPort As Byte, ByVal bytAddress As Byte, ByVal bytAngleOutputRange As Byte) As Long
```

Example VB Usage:

```
Dim bytComPort As Byte
Dim bytAddress As Byte
Dim IResult As Long
Dim bytAngleOutputRange As Byte
bytComPort = 1 ' Access COM1...
bytAddress = 1
IResult = T7_GetAngleOutputRange(bytComPort, bytAddress, bytAngleOutputRange)
If IResult <> 0 Then
    ' Handle error...
End If
```

5.17 T7_SetAngleOutputRange

Description:

The T7_SetAngleOutputRange function is used to set the angle output range for all three axes to bidirectional (-180.000° to 179.999°) or unidirectional (0.000° to 359.999°)

Passed Parameters:



ucComPort

Specifies the COM port number to be accessed. Valid range is from 1 to 255.

ucAddress

Specifies the devices address. Valid range is from 1 to 127.

ucAngleOutputRange

0 = bidirectional (-180.000° to 179.999°)

1 = unidirectional (0.000° to 359.999°)

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall T7_SetAngleOutputRange(unsigned char ucComPort, unsigned char ucAddress,  
unsigned char ucAngleOutputRange);
```

Example C Usage:

```
int iResult = 0;  
unsigned char ucComPort = 1;  
unsigned char ucAddress = 1;  
unsigned char ucAngleOutputRange = 1;  
iResult = T7_SetAngleOutputRange(ucComPort, ucAddress, ucAngleOutputRange);  
if (iResult != 0) {  
    // Handle error...  
}
```

VB Declaration:

```
Public Declare Function T7_SetAngleOutputRange Lib "T7User.dll" (ByVal bytComPort As Byte, ByVal  
bytAddress As Byte, ByVal bytAngleOutputRange As Byte) As Long
```

Example VB Usage:

```
Dim bytComPort As Byte  
Dim bytAddress As Byte  
Dim IResult As Long  
Dim bytAngleOutputRange As Byte  
bytAngleOutputRange = 1  
bytComPort = 1 ' Access COM1...  
bytAddress = 1  
IResult = T7_SetAngleOutputRange(bytComPort, bytAddress, bytAngleOutputRange)  
If IResult <> 0 Then  
    ' Handle error...  
End If
```

5.18 T7_GetDeviceInfo

Description:

The T7_GetDeviceInfo function gets the following factory settings from the T7: serial number, firmware revision, product type and calibration status.



Passed Parameters:

ucComPort

Specifies the COM port number to be accessed. Valid range is from 1 to 255.

ucAddress

Specifies the devices address. Valid range is from 1 to 127.

pulSerialNo

Pointer to a long value that will hold the returned serial number.

puiFirmware

Pointer to a 6 byte array of characters that will hold the returned firmware version. Padded with spaces and not null terminated.

puiProduct

Pointer to a 6 byte array of characters that will hold the returned product type. Padded with spaces and not null terminated.

puiCalibrationStatus

Pointer to an unsigned integer describing the calibration status:

Bit	Description
0	set if axis 0 is calibrated
1	set if axis 1 is calibrated
2	set if axis 2 is calibrated
3	set if calibration is temperature compensated
4-15	reserved

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall T7_GetDeviceInfo(unsigned char ucComPort, unsigned char ucAddress, unsigned long * pulSerialNo, unsigned char * pucFirmware, unsigned char * pucProduct, unsigned short * puiCalibrationStatus);
```

Example C Usage:

```
int iResult = 0;
unsigned char ucComPort = 1;
unsigned long ulSerialNo = 0;
unsigned char ucAddress = 1;
unsigned char ucFirmware[6];
unsigned char ucProduct[6];
unsigned short uiCalibrationStatus = 0;
iResult = T7_GetDeviceInfo(ucComPort, ucAddress, &ulSerialNo, &ucFirmware, &ucProduct, &uiCalibrationStatus);
if (iResult != 0) {
    // Handle error...
}
```

VB Declaration:



US Digital • 1400 NE 136th Avenue • Vancouver, Washington • 98684 • USA •
Local: 360-260-2468 • Toll-free: 800-736-0194 • Support: 360-397-9999 • Fax: 360-260-2469
Email: info@usdigital.com • Website: www.usdigital.com

Public Declare Function T7_GetDeviceInfo Lib "T7User.dll" (ByVal *bytComPort* As Byte, ByVal *bytAddress* As Byte, ByRef *lSerialNo* As Long, ByRef *bytFirmware* As Byte, ByRef *bytProduct* As Byte, ByRef *iCalibrationStatus* As Integer) As Long

Example VB Usage:

```
Dim bytComPort As Byte
Dim bytAddress As Byte
```

```
Dim lResult As Long
Dim lSerialNo As Long
Dim bytFirmware(0 To 5) As Byte
Dim bytProduct(0 To 5) As Byte
Dim iCalibrationStatus As Integer
bytComPort = 1 ' Access COM1...
bytAddress = 1
lResult = T7_GetDeviceInfo(bytComPort, bytAddress, lSerialNo, bytFirmware, bytProduct,
iCalibrationStatus)
If lResult <> 0 Then
    ' Handle error...
End If
```

5.19 T7_PingAddress

Description:

The T7_PingAddress function causes all devices attached to the US Digital CAN Adapter to report their current address, device type, and serial number.

Passed Parameters:

ucComPort

Specifies the COM port number to be accessed.

ucAddress

Specifies the devices address. 0 = Reserved for CAN adapter, Valid range is 1 to 127. The default address for an T7 device is 127. All T7's listen to address 126.

pucCurrentAddress

Pointer to an array of unsigned chars that will hold the address of each device that responded to the ping.

pucDeviceType

Pointer to an array of unsigned chars that will hold the device type of each device that responds to the ping. The T7-3 is device type 1 and T7-1 is device type 4. The CAN adapter is device type 2.

pulSerialNo

Pointer to an array of unsigned longs that will hold the serial number of each device that responds to the ping.

pucSize

Pointer to a byte that specifies the size of each parameter array and returns the number of elements copied.

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.



US Digital • 1400 NE 136th Avenue • Vancouver, Washington • 98684 • USA •
Local: 360-260-2468 • Toll-free: 800-736-0194 • Support: 360-397-9999 • Fax: 360-260-2469
Email: info@usdigital.com • Website: www.usdigital.com

C Declaration:

```
extern int __stdcall T7_PingAddress(unsigned char ucComPort, unsigned char ucAddress,  
unsigned char * pucCurrentAddress, unsigned char * pucDeviceType, unsigned long * pulSerialNo,  
unsigned char * pucSize);
```

Example C Usage:

```
int iResult = 0;  
unsigned char ucCOMPort = 1;  
unsigned char ucAddress = 126; // Ping All T7's...  
unsigned char ucCurrentAddress[16];  
unsigned char pucDeviceType [16];  
unsigned long pulSerialNo [16];  
unsigned char ucSize = 16;  
iResult = T7_PingAddress(ucComPort, ucAddress, ucCurrentAddress, pucDeviceType, pulSerialNo,  
&ucSize);  
if (iResult != 0) {  
    // Handle error...  
}
```

VB Declaration:

```
Public Declare Function T7_PingAddress Lib "T7User.dll" (ByVal bytComPort As Byte, ByVal bytAddress  
As Byte, ByRef bytCurrentAddress As Byte, ByRef bytDeviceType As Byte,  
ByRef ISerialNo As Long, ByRef bytSize As Byte) As Long
```

Example VB Usage:

```
Dim bytComPort As Byte  
Dim bytAddress As Byte  
Dim IResult As Long  
Dim bytCurrentAddress(0 To 15) As Byte  
Dim bytDeviceType(0 To 15) As Byte  
Dim ISerialNo(0 To 15) As Long  
Dim bytSize As Byte
```

```
bytComPort = 1 ' Access COM1...  
bytAddress = 126 ' Ping all T7's...  
bytSize = 16  
IResult = T7_PingAddress(bytComPort, bytAddress, bytCurrentAddress(0), bytDeviceType(0),  
ISerialNo(0), bytSize)  
If IResult <> 0 Then  
    ' Handle error...  
End If
```

5.20 T7_SetAddress

Description:

The T7_SetAddress function changes the address of an T7.

Passed Parameters:

ucComPort



US Digital • 1400 NE 136th Avenue • Vancouver, Washington • 98684 • USA •
Local: 360-260-2468 • Toll-free: 800-736-0194 • Support: 360-397-9999 • Fax: 360-260-2469
Email: info@usdigital.com • Website: www.usdigital.com

Specifies the COM port number to be accessed.

ucAddress

Specifies the devices address. 0 = Reserved for CAN adapter, Valid range is 1 to 127. The default address for an T7 device is 127. All T7's listen to address 126.

ucDeviceType

An unsigned char that identifies the device type. The device type for an T7-3 is 1 and T7-1 is 4.



US Digital • 1400 NE 136th Avenue • Vancouver, Washington • 98684 • USA •
Local: 360-260-2468 • Toll-free: 800-736-0194 • Support: 360-397-9999 • Fax: 360-260-2469
Email: info@usdigital.com • Website: www.usdigital.com

ulSerialNo

Unsigned long value that matches the T7 device's serial number whose address is to be changed.

ucNewAddress

The new address that will be written to the T7 device.

Returns:

Result code as 32-bit integer: See error code section for values other than zero. Zero implies the function call is successful.

C Declaration:

```
extern int __stdcall T7_SetAddress (unsigned char ucComPort, unsigned char ucAddress,  
unsigned char ucDeviceType, unsigned long ulSerialNo, unsigned char ucNewAddress);
```

Example C Usage:

```
int iResult = 0;  
unsigned char ucCOMPort = 1;  
unsigned char ucAddress = 127;  
unsigned char ucDeviceType = 1;  
unsigned long ulSerialNo = 12345;  
unsigned char ucNewAddress = 1;  
iResult = T7_SetAddress(ucComPort, ucAddress, ucDeviceType, ulSerialNo, ucNewAddress);  
if (iResult != 0) {  
    // Handle error...  
}
```

VB Declaration:

```
Public Declare Function T7_SetAddress Lib "T7User.dll" (ByVal bytComPort As Byte, ByVal bytAddress  
As Byte, ByVal bytDeviceType As Byte, ByVal lSerialNo As Long, ByVal bytNewAddress As Byte) As Long
```

Example VB Usage:

```
Dim lResult As Long  
Dim bytComPort As Byte  
Dim bytAddress As Byte  
Dim bytDeviceType As Byte  
Dim lSerialNo As Long  
Dim bytSize As Byte  
  
bytComPort = 1 ' Access COM1...  
bytAddress = 1  
ulSerialNo = 12345  
bytNewAddress = 1  
lResult = T7_SetAddress(bytComPort, bytAddress, bytDeviceType, lSerialNo, bytNewAddress)  
If lResult <> 0 Then  
    ' Handle error...  
End If
```



6 Constants

```
#define T7_MAX_COMM_PORTS          255 // Maximum number of COM ports
#define T7_MAX_DEVICES_PER_PORT    64  // Maximum number of device per COM
                                     // port or CAN bus
#define T7_MAX_BAUD_RATE          115200 // Maximum baud rate
                                     // Supported baud rates: 9600, 19200,
                                     // 38400, 57600, 115200
#define T7_MIN_DAMPING_CODE        2   // Minimum damping time (millisecond)
#define T7_MAX_DAMPING_CODE        5000 // Maximum damping time (millisecond)
#define T7_BROADCAST_ADDRESS       126 // CAN bus address that all devices
                                     // listen to
```

7 Error Codes

```
#define T7_SUCCESS                  0x00 // Success.
#define T7_INVALID_COMMAND          0x01 // Invalid command.
#define T7_RESERVED_1              0x02 // Reserved
#define T7_INVALID_PARAMETER        0x03 // Invalid Parameter.
#define T7_CHECKSUM_ERROR_TX        0x04 // Checksum error sent to the T7.
#define T7_COMMAND_FAILED          0x05 // Command failed.
#define T7_RESERVED_2              0x06 // Reserved
#define T7_FLASH_ERASE_ERROR        0x07 // Flash erase error from T7.
#define T7_FLASH_PROGRAM_ERROR      0x08 // Flash program error from T7.
#define T7_ADDRESS_OUT_OF_RANGE     0x09 // Address out of Range

#define T7_INVALID_COMPORT          0x80 // Invalid com port specified.
#define T7_FAIL_TO_OPEN_COM_PORT    0x81 // Failed to com port.
#define T7_COMM_ERROR               0x82 // Generic com error.
#define T7_COM_PORT_NOT_OPEN        0x83 // Com port not open.
#define T7_FAILED_TO_SET_COM_PORT_TIMEOUT 0x84 // Failed to set com port timeout.
#define T7_FAILED_TO_FLUSH_COM_PORT 0x85 // Failed to purge the com port.
#define T7_FAILED_TO_CLEAR_COM_ERROR 0x86 // Failed to clear com port error.
#define T7_INVALID_AXIS             0x87 // Invalid axis specified.
#define T7_COMM_TIMEOUT             0x88 // Failed to receive expected data.
#define T7_CHECKSUM_ERROR_RX        0x89 // Checksum error sent from the T7.
```



8 Enumerations

C Declaration	VB Declaration
<pre>enum eTemperatureRange { eLow = 0, eMid = 1, eHigh = 2 };</pre>	<pre>Public Enum eAxis eXAxis = 0 ePitch = 0 eYAxis = 1 eRoll = 1 eZAxis = 2 eYaw = 2 End Enum</pre>
<pre>enum eTemperatureRange { eLow = 0, eMid = 1, eHigh = 2 };</pre>	<pre>Enum eTemperatureRange eLow = 0 eMid = 1 eHigh = 2 End Enum</pre>
<pre>enum eBaudRateCode { e115200 = 0, e57600 = 1, e38400 = 2, e19200 = 3, e9600 = 4 };</pre>	<pre>Enum eBaudRateCode e115200 = 0 e57600 = 1 e38400 = 2 e19200 = 3 e9600 = 4 End Enum</pre>

